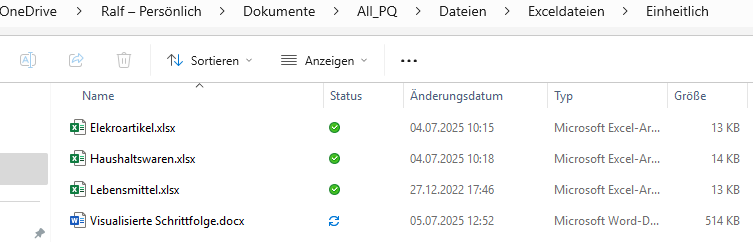
Aufgabe: Alle Tabellenblätter Namens Tabelle1 aus allen Exceldateien eines Ordners einlesen und zusammenführen.

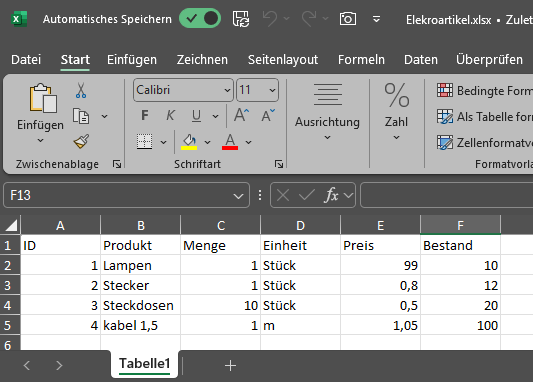
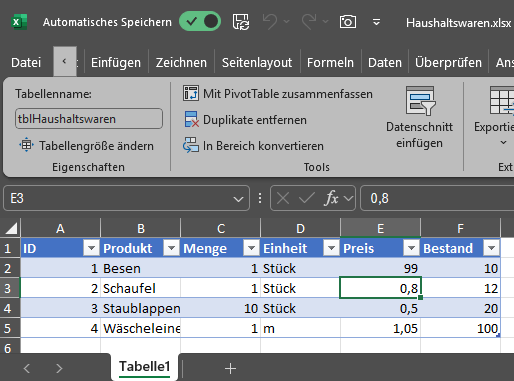
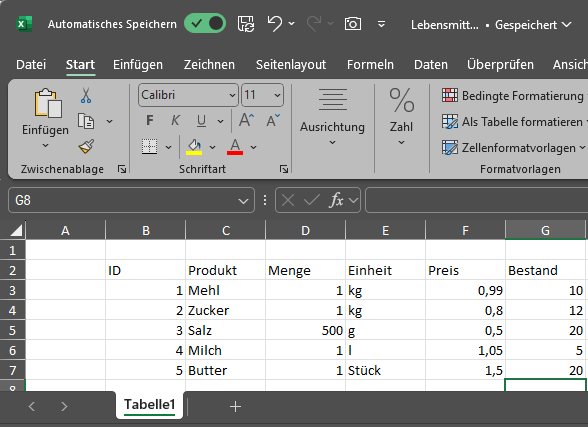
Ausgangslage: Im Ordner



…befinden sich 3 Exceldateien mit gleichartiger Struktur, sowie ein Worddokument:

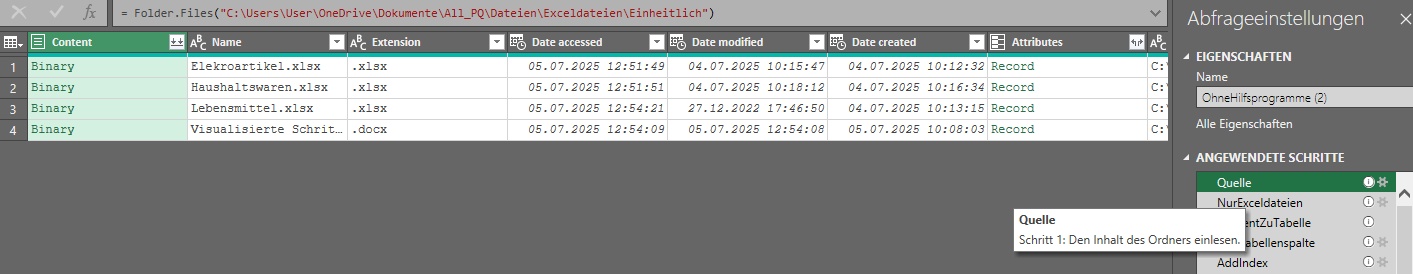
Die Exceldateien:

Elektroartikel.xlsx Haushaltwaren.xlsx Lebensmittel.xlsx

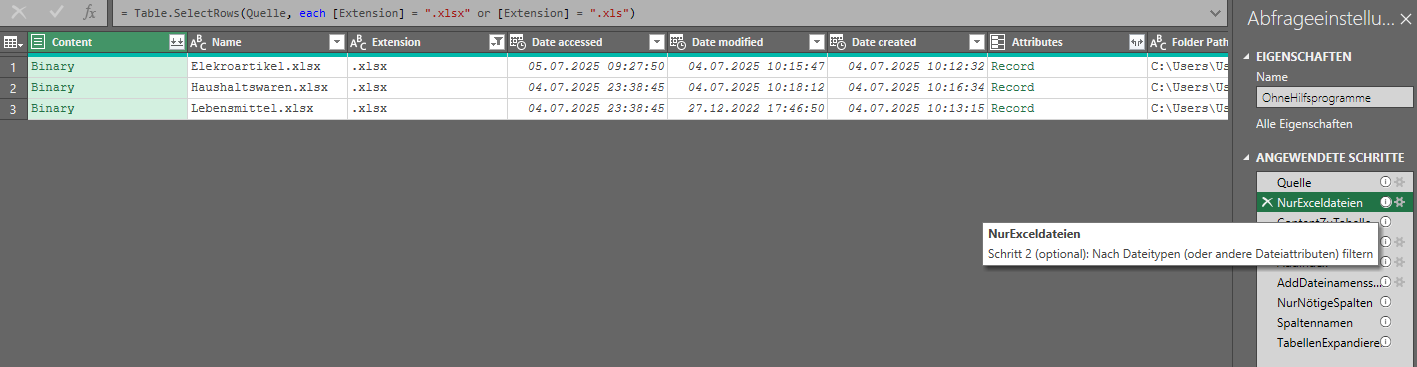
Allen gemeinsam ist, dass sich alle Daten im Arbeitsblatt (Sheet) ***Tabelle1*** befinden. Auch, wenn sich im Workbook ***Haushaltwaren.xlsx*** die Daten in einer *formatierten Tabelle* Namens ***tblHaushaltwaren*** befinden. Die Struktur (Spaltenanzahl und Spaltennamen) sowie der Ort (Tabelle1) ist bei allen identisch. Auch wenn der Datenbereich der ***Lebensmittel.xlsx*** sowohl in Größe (Anzahl der Datensätze) als auch in der Position abweicht.

1. Schritt: ***Quelle = Folder.Files(„Ordnerpfad“)***



1. Schritt: ***NurExceldateien = Table.SelectRows(Quelle, each [Extension] = ".xlsx" or [Extension] = ".xls"),***

Nur Dateien mit der Endung .*xlsx* oder .*xls* zulassen.



1. Schritt: **ContentZuTabelle = Table.FromList(NurExceldateien[Content], Splitter.SplitByNothing(), null, null, ExtraValues.Error),**

Die Liste der Spalte *Content* in eine Tabelle umwandeln



1. Schritt: Der wird etwas komplexer, da jede Zeile der neu zu erstellenden Spalte mehrere Bearbeitungsschritte benötigt.  
   Deshalb werden die **in eine eigene Abfrage gepackt**:

Die gesamte Anweisung des 4. Schrittes lautet:

***AddTabellenspalte*** *= Table.AddColumn(ContentZuTabelle,"Datei transformieren", each*

***let***

***Tab = Excel.Workbook([Column1], null, true),***

***Tabelle1\_Sheet = Tab{[Item="Tabelle1",Kind="Sheet"]}[Data],***

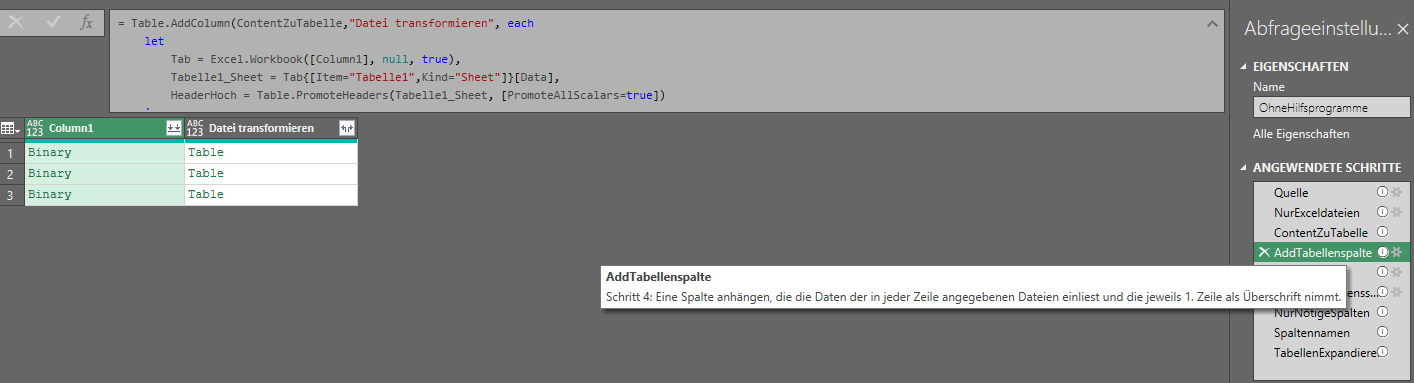
***HeaderHoch = Table.PromoteHeaders(Tabelle1\_Sheet, [PromoteAllScalars=true])***

***in***

***HeaderHoch***

***),***

Es wird also eine Spalte Namens ***Datei transformieren*** erzeugt, die die Datentabellen der in ***Column1*** angegebenen Workbooks enthält.



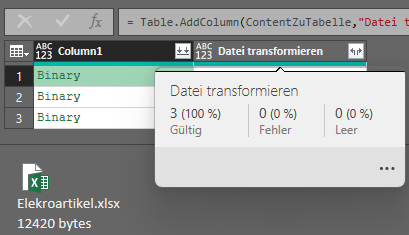
Der 3. und letzte Parameter der Table.AddColumn – Funktion erwartet eine Funktion zur Generierung des Spalteninhalts. Eine Funktion kann mit ***each*** oder einem Klammernpaar (\_) => eingeleitet werden.

Eine Funktion als Ergebnis wäre für uns allerdings nicht hilfreich, da wir Tabellen für die spätere Auswertung benötigen.

Das Schlüsselwort each leitet dennoch unsere Absicht ein. Der Teil der separaten Abfrage ist der oben gelb hinterlegte. Schauen wir uns die nach ***each*** folgenden Einzelanweisungen etwas genauer an, die mit dem Schlüsselwort let eingeleitet werden.

***Tab = Excel.Workbook([Column1], null, true),***

Die Funktion ***Excel.Workbook(…)*** gibt eine Arbeitsmappe zurück. Wobei der 1. Parameter vom Typ ***Binary*** sein muss. Und genau das finden wir in der Spalte ***Column1***. Wenn wir in den freien Bereich einer beliebige Zelle von ***Column1*** klicken, können wir unten in der Ausgabevorschau (hier rot umrandet) sehen, um welche Datei es sich handelt.

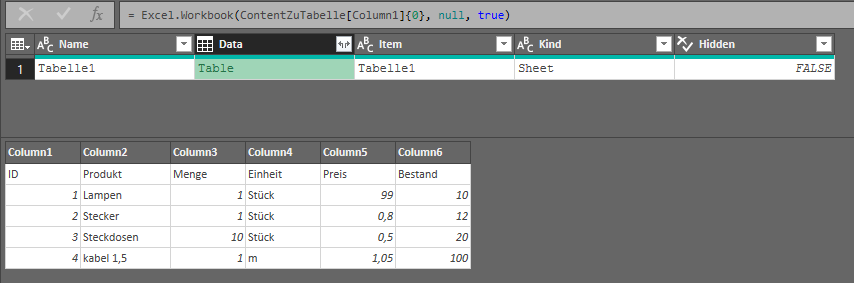


Die 2. Anweisung: ***Tabelle1\_Sheet = Tab{[Item="Tabelle1",Kind="Sheet"]}[Data],*** weist das Programm an**,** dass die Daten aus allen Tabellenblättern (***Sheet***) mit dem Namen ***Tabelle1*** in der Spalte ***Data*** ausgelesen werden sollen.

Damit Du Dir das vorstellen kannst, schiebe ich mal die Anweisung dazwischen, mir die Daten des 1. Datensatzes (Achtung! Null-basiert) des Abfrageschrittes ***ContentZuTabelle*** anzeigen zu lassen. **Diese Anweisung ist NICHT Teil unserer Abfrage!**

Wie gut zu erkennen ist, gibt es dort eine Spalte Namens ***Data***, deren Datenvorschau genau das liefert, was wir zu sehen wünschen.

Bei der Gelegenheit will ich gleich darauf aufmerksam machen, dass (in diesem Fall) die eigentlichen Überschriften in Zeile 1 stehen. Das muss nicht immer der Fall sein, aber da es nun einmal so ist, werden wir im nächsten Schritt darauf reagieren.



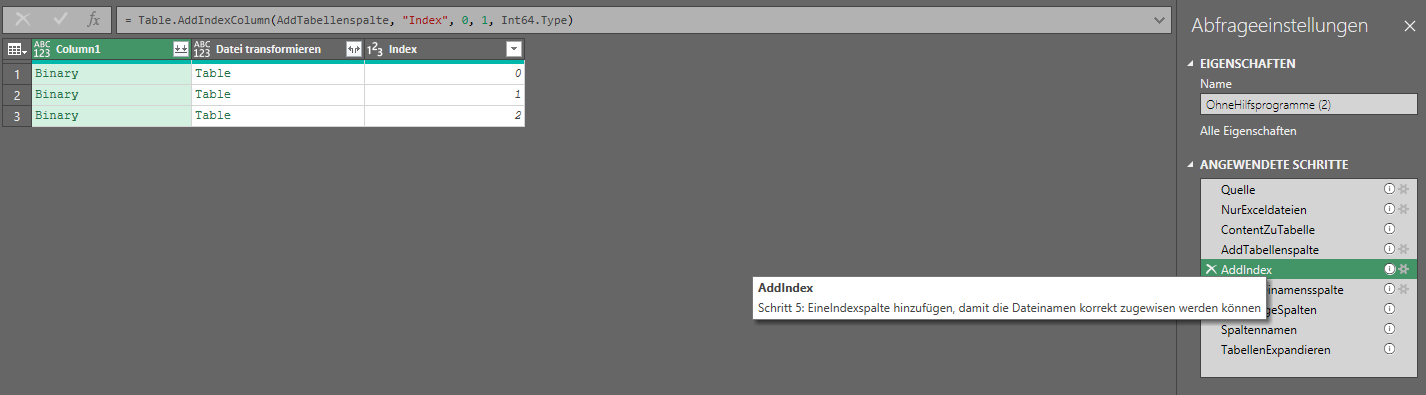
Denn der nächste und letzte Schritt unserer separaten Abfrage lautet:

***HeaderHoch = Table.PromoteHeaders(Tabelle1\_Sheet, [PromoteAllScalars=true])***

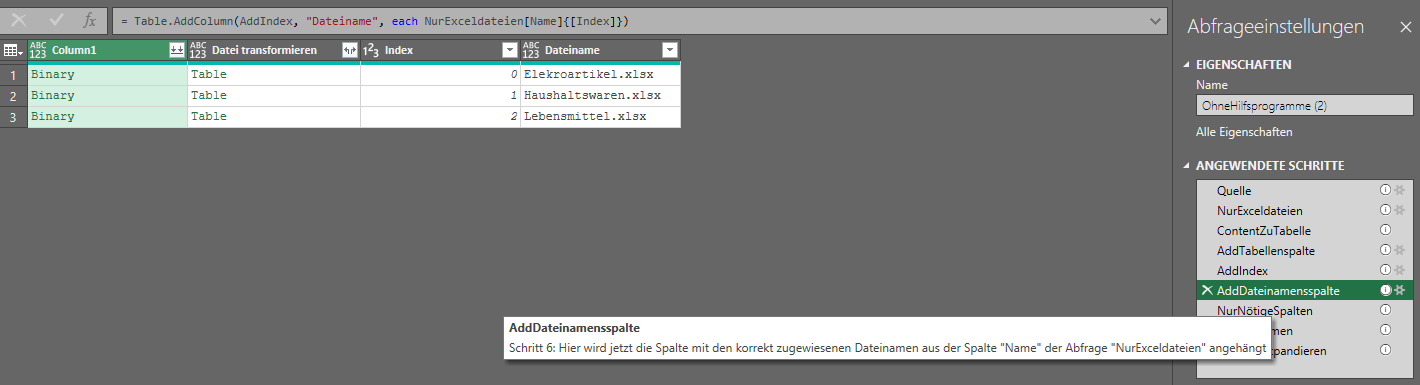
…und bedeutet nichts anderes als die Werte der 1. Zeile als Überschriften zu verwenden….

Der letzte Schritt „***HeaderHoch***“ ist auch der, der in jede (each) Zeile der neuen Spalte ausgegeben werden soll.

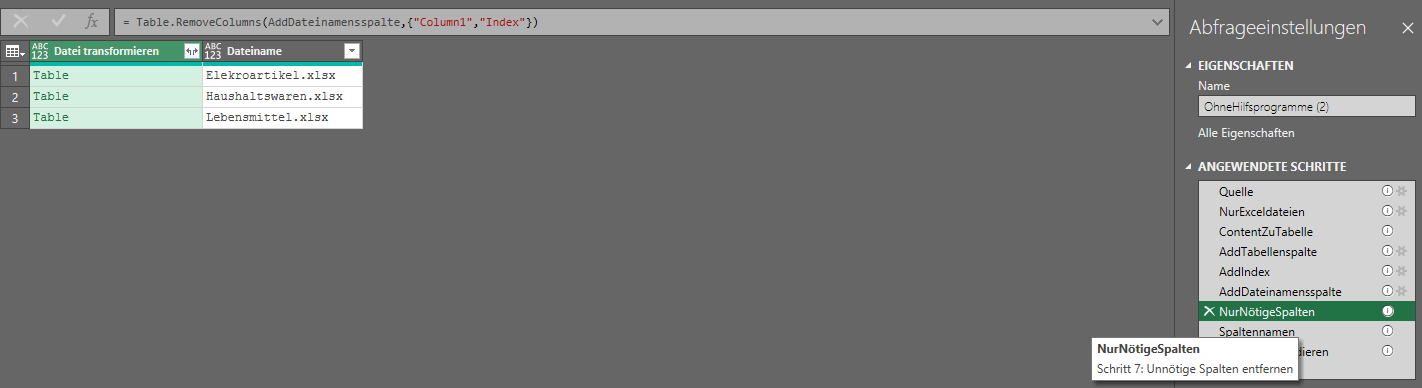
1. Schritt: Dieser Schritt ist nur deshalb notwendig, weil wir Werte zweier voneinander unabhängiger Tabellen miteinander in Verbindung bringen wollen. Das wären zum einen die Dateinamen, die wir aus Schritt 2 (nur Exceltabellen anzeigen) nehmen und in unser letztes Abfrageergebnis übertragen wollen. Das geht hier, weil die Tabellen der Spalte ***Datei transformieren*** noch nicht expandiert wurden. In beiden Fällen haben wir 3 Datensätze. Damit sie entsprechend zugeordnet werden können, müssen wir wissen, in welchen Zeilen sie stehen. Dazu benötigen wir kurzzeitig eine Indexspalte.  
     
   ***AddIndex = Table.AddIndexColumn(AddTabellenspalte, "Index", 0, 1, Int64.Type),***  
     
   Hier geben wir an, dass die Spalte die Überschrift ***Index*** hat, mit ***0*** beginnt und in ***Einer-Schritten*** fortgeführt werden soll.

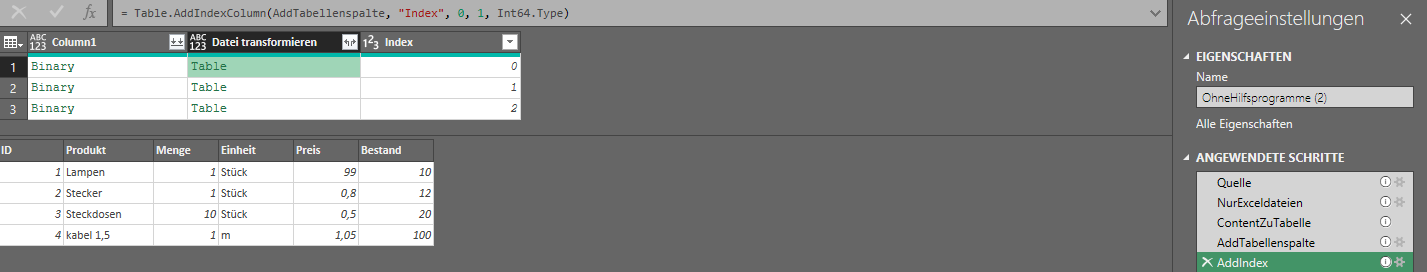


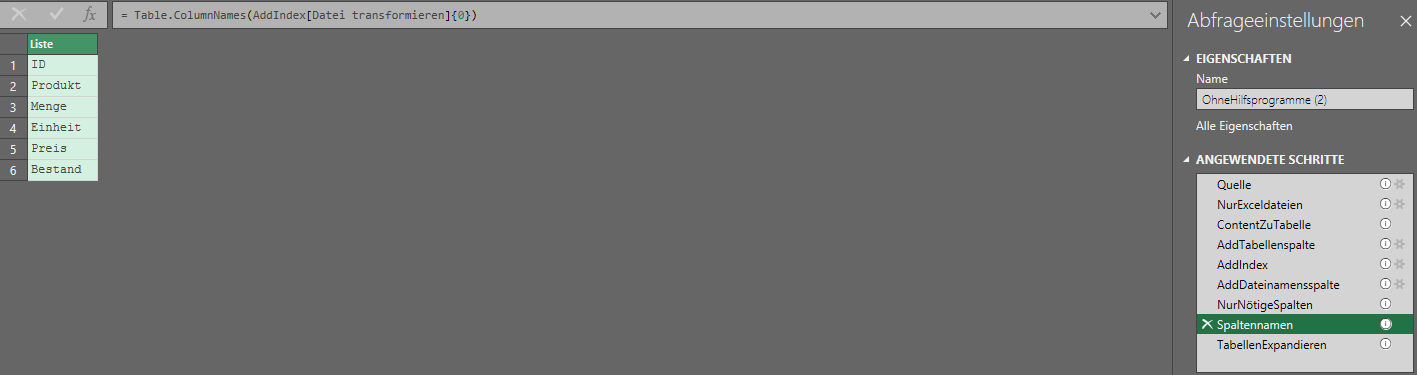
1. Schritt: ***AddDateinamensspalte = Table.AddColumn(AddIndex, "Dateiname", each NurExceldateien[Name]{[Index]}),***Entscheidend ist der gelb unterlegte Teil. Sieh im Schritt ***NurExceldateien*** in der Spalte ***[Name]*** in der ***{Zeile nach, die in der Spalte [Index]}*** steht. Wobei die Zeilenangabe immer in **{ }** und die Spaltenangabe in **[ ]** steht. Die Zeile, die Spalte Index steht ist somit:  
     
   **{[**Index**]}**Eine direkte Zeilenangabe für Zeile 4 wäre: **{3}**



1. Schritt: ***NurNötigeSpalten = Table.RemoveColumns(AddDateinamensspalte,{"Column1","Index"}),***  
   Schnell erklärt: Alle nicht benötigten Spalten (in diesem Fall ***Column1*** und ***Index***). Nicht benötigte Spalten (auch wenn es nur eine ist) werden als Liste in **{ }** und Anführungszeichen angegeben. Sind es mindestens 2 Spalten, werden sie kommasepariert,



1. Schritt: ***Spaltennamen = Table.ColumnNames(AddIndex[Datei transformieren]{0}),***  
   Diesen Schritt fügen wir deshalb ein, um dynamischer auf Änderungen im nachfolgenden Schritt reagieren zu können. Egal, ob der Pfad geändert wird, oder sich die Spaltenbezeichnungen und/oder deren Anzahl ändert, es wird immer eine Liste mit den Überschriften geben, die ***im 1. Datensatz*** (nullbasiert) des Abfrageschrittes ***AddIndex*** in der Spalte ***Datei transformieren*** zu finden sind. Und das wären diese:  
     
     
     
   Zum Vergleich – Das eigentliche Abfrageergebnis:



1. Schritt:  ***TabellenExpandieren = Table.ExpandTableColumn(NurNötigeSpalten, "Datei transformieren",Spaltennamen,Spaltennamen)***

Zu guter Letzt werden die in der Spalte ***Datei transformieren*** der Abfrage ***NurNötigeSpalten*** enthaltenen Tabellen expandiert.

Dabei wird die im Vorgängerschritt ermittelte Liste der Spaltennamen für die letzten beiden Parameter der Funktion ***Table.ExpandTableColumn*** verwendet.

